Semiring Rank Matrix Factorization

Thanh Le Van, Siegfried Nijssen, Matthijs van Leeuwen, Luc de Raedt

Semiring Rank Matrix Factorisation

Thanh Le Van, Siegfried Nijssen, Matthijs van Leeuwen, Luc De Raedt

Abstract—Rank data, in which each row is a complete or partial ranking of available items (columns), is ubiquitous. Among others, it can be used to represent preferences of users, levels of gene expression, and outcomes of sports events. It can have many types of patterns, among which consistent rankings of a subset of the items in multiple rows, and multiple rows that rank the same subset of the items highly. In this article, we show that the problems of finding such patterns can be formulated within a single generic framework that is based on the concept of semiring matrix factorisation. In this framework, we employ the max-product semiring rather than the plus-product semiring common in traditional linear algebra. We apply this semiring matrix factorisation framework on two tasks: sparse rank matrix factorisation and rank matrix tiling. Experiments on both synthetic and real world datasets show that the framework is capable of discovering different types of structure as well as obtaining high quality solutions.

Index Terms—Rank data, rank matrix factorisation, pattern set mining, rank matrix tiling, integer programming, semiring, max-product.

1 Introduction

We develop a generic framework for unsupervised discovery of regularities (patterns) in *rank data*. In this type of data, each row (transaction) is a complete or a partial ranking of the available columns (items). Rank data naturally occurs in many situations of interest. Consider, for instance, cycling competitions where the items are the cyclists and each transaction corresponds to a race, or a business context, where the items are companies and the transactions specify the rank of their quotation for a particular service. In social sciences, rank data has been used to represent users' preferences over their favorite countries [1], presidency candidates [2], [3] or products [4]. In biology, rank data has been used to represent the levels of gene expression [5], [1]. In sport analytics, rank data has been used to rank sport teams [6]. In general, ranking forms a natural abstraction for purely numeric data, which often arises in practice and may be noisy or imprecise. Especially when the rows are incomparable, i.e., when they contain measurements on different scales, transforming the data to rankings may result in a more informative representation [1], [7], [8].

While rank data is ubiquitous, only few data mining methods have been developed for rank data analysis. Exceptions include the work by Ben-Dor et al., [5], who proposed a probabilistic model to discover a fix-sized order-preserving rectangle; and the work by Henzgen et al., [9], who proposed an algorithm to enumerate frequent order-preserving items. We also contributed a rank matrix tiling method [1] to discover *ranked tiles*, which are data rectangles having high ranks. Each of these works aimed at a single type of rank pattern and they did not aim at a general framework for different types of rank pattern set mining, i.e., a small, non-redundant set of patterns globally describing the structure of the data [10].

Matrix factorisation has been used in many fields such as data mining [11], [12], recommender systems [13] and bioinformatics [14]. Depending on the constraints on the data or the patterns users are interested in, one applies different forms of matrix factorisation. For example, if the given data has non-negative value constraints, non-negative matrix factorisation [15] can be employed; if users are interested in sparse features, sparse dictionary learning [16] can be considered. Although matrix factorisation has been extensively studied, it cannot be applied directly to rank data due to the fact that the linear algebra used in the traditional matrix factorisation methods does not provide a way to aggregate/sum rankings over items (see Section 2).

Another class of methods that have been developed to find patterns in numerical data are biclustering methods [17], which are particularly popular in bioinformatics; however, biclustering algorithms for the rank data settings studied in this article do not currently exist either.

Our contributions can be summarised as followed. First, we introduce a generic Semiring Rank Matrix Factorisation framework named *sRMF* for mining sets of patterns in rank data. Second, we show that the *sRMF* framework generalises our conference papers [1] and [7]. In [7] we introduced rank matrix factorisation as a model to mine rank pattern sets, and in [1] we studied tiling in rank data. Using the semiring abstraction, both problems can be studied within the same generic framework. This does not only lead to a more general framework; it also leads to improved performance.

The rest of the paper is organised as follows. We introduce the sRMF framework in Section 2. Then, we demonstrate how to apply the framework on the two problem instances, including Sparse RMF [7] and rank matrix tiling [1], in Sections 3 and 4 respectively. Experiments are presented in Section 6 and 7. We discuss related work in Section 8 and

- *Thanh Le Van is with INRIA, MAGNET team, Lille 59650, France.*
- *Siegfried Nijssen is with the Institute of Information and Communication Technologies, Electronics and Applied Mathematics of the Université catholique de Louvain, Belgium.*
- *Matthijs van Leeuwen is with the Leiden Institute for Advanced Computer Science, Universiteit Leiden, The Netherlands.*
- *Luc De Raedt is with the Department of Computer Science, KU Leuven, Belgium.*

Manuscript received ? 2016; revised ?, ?.

conclude in Section 9.

2 SEMIRING RANK MATRIX FACTORISATION (SRMF)

In this section, we first illustrate the rank pattern set mining problem. Next, we explain the reason why the traditional matrix factorisation approaches based on linear algebra cannot be directly used for mining rank data. Then, we introduce the semiring rank matrix factorisation framework.

Definition 1 (Rank matrix). An $m \times n$ matrix $\mathbf{M}$ is a rank matrix iff $\mathbf{M}_{r,c} \in \sigma$, for all $1 \leq r \leq m$ and $1 \leq c \leq n$, where $\sigma = \{1, 2, ..., n\} \cup \{0\}$.

In our setting, columns are items or products that need to be ranked; rows are rankings of items. Matrix entry $\mathbf{M}_{r,c}$ indicates that column c is ranked $\mathbf{M}_{r,c}$th for row r. The rank value 0 has a special meaning. It denotes *unknown* rankings. For example, in rating datasets, some items are not rated. Such items will have rank value 0.

Many different types of patterns can exist in rank matrices. We will first discuss the intuitions behind two such pattern types.

Example 1 (Consistent Ranks). Consider the following rank matrix:

$$\mathbf{M} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 2 & 3 & 4 & 5 & 6 \\ 2 & 3 & 5 & 6 & 4 & 1 \\ 2 & 3 & 5 & 6 & 1 & 4 \end{pmatrix}$$

In red and blue we indicated parts of the matrix in which the rank is consistent for a subset of rows and columns of the matrix: for instance, in the first two rows, the ranks of the items are identical. Red and blue here highlight patterns in the matrix.

Example 2 (High Ranks). Consider the following rank matrix:

$$\mathbf{M} = \begin{pmatrix} 1 & 2 & 5 & 4 & 6 & 3 \\ 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 3 & 2 & 4 & 5 & 6 \\ 2 & 3 & 5 & 6 & 4 & 1 \\ 2 & 3 & 5 & 6 & 1 & 4 \end{pmatrix}$$

In red and blue we indicated subsets of columns and rows in which the ranks are greater than 3 (the average rank). These subsets of rows and columns point towards patterns in the data; while the rank within these patterns may be consistent, this is not necessarily the case, as illustrated by the red pattern.

The aim of this article is to present a generic framework that is expressive and flexible enough to model and discover small sets of these different types of rank patterns. Our main observation is that finding such small sets of patterns can be formalised as a rank factorisation problem.

Definition 2 (Rank matrix factorisation). Given a rank matrix $\mathbf{M} \in \sigma^{m \times n}$ and an integer k, find a matrix $\mathbf{C}^* \in \{0,1\}^{m \times k}$ and a matrix $\mathbf{F}^* \in \sigma_p^{k \times n}$ such that:

$$(\mathbf{C}^*, \mathbf{F}^*) \equiv \underset{\mathbf{C},\mathbf{F}}{\operatorname{argmax}}\ f(\mathbf{M}, \mathbf{C} \odot \mathbf{F}). \tag{1}$$

where

- $f(,)$ is a scoring function that measures the similarity between matrices;
- $\odot$ is an operator that creates a data matrix based on two factor matrices;
- $\sigma_p \subseteq \sigma$ is a set of permissible values in σ.

Intuitively, in matrix $\mathbf{F}$ the rows $\mathbf{F}_{i,:}$ indicate partial rankings. Columns $\mathbf{C}_{:,i}$ of matrix $\mathbf{C}$ indicate in which rows the corresponding partial ranking appears. The following example illustrates this intuition for Example 1.

Example 3 (Rank matrix factorisation). The patterns for Example 1 can be represented as follows using two matrices $\mathbf{C}$ and $\mathbf{F}$:

$$\mathbf{C} = \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{pmatrix} \qquad \mathbf{F} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 2 & 3 & 5 & 6 & 0 & 0 \end{pmatrix}$$

This factorisation summarises matrix $\mathbf{M}$ with two rank vectors: one is the *full rank* vector $u = (1, 2, 3, 4, 5)$, which appears in row 1 and row 2 of the matrix, and the other is the *partial rank* vector $v = (2, 3, 5, 6, 0, 0)$, which appears in the last two rows.

A first important choice that needs to be made in this framework concerns the choice for the operator $\odot$. An obvious choice for this operator may be to use the traditional matrix product. However, this choice causes problems.

Example 4 (Overlapping rank profiles).

$$\begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 1 \\ 0 & 1 \\ 0 & 1 \end{pmatrix} \times \begin{pmatrix} 1 & 2 & 5 & 6 & 0 & 0 \\ 0 & 0 & 4 & 6 & 1 & 2 \end{pmatrix} = \begin{pmatrix} 1 & 2 & 5 & 6 & 0 & 0 \\ 1 & 2 & 9 & 12 & 1 & 2 \\ 1 & 2 & 9 & 12 & 1 & 2 \\ 0 & 0 & 4 & 6 & 1 & 2 \\ 0 & 0 & 4 & 6 & 1 & 2 \end{pmatrix}$$

The factorisation in this example says that the two partial rank profiles are both present in row 2 & 3. Using the normal matrix product, the combined rankings for both row 2 and 3 become $v = (1, 2, 9, 12, 1, 2)$. This is an invalid rank vector as it violates the definition of a rank matrix (Definition 1), which requires values in each row to belong to σ.

For this reason, we require a different choice for the $\odot$ operator. In this article, we will consider operators that are based on *semirings* [18] to ensure that the output of a matrix product remains within the range of valid ranks.

Definition 3 (Semiring). A semiring $(\sigma, \oplus, \otimes)$ is a set σ equipped with two binary operations $\oplus$ and $\otimes$ satisfying the following properties:

- $\oplus$ is commutative: $a \oplus b = b \oplus a$;
- $\otimes$ and $\oplus$ are associative: $a \otimes (b \otimes c) = (a \otimes b) \otimes c$, $a \oplus (b \oplus c) = (a \oplus b) \oplus c$;
- σ has identity elements for $\oplus$ and $\otimes$, indicated with 0 and 1, such that $a \otimes 1 = a$, $1 \otimes a = a$, and $a \oplus 0 = a$;
- $\otimes$ left and right distributes over $\oplus$: $a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c)$, $(a \oplus b) \otimes c = (a \otimes c) \oplus (b \otimes c)$;
- the 0 element annihilates for all elements in σ: $0 \otimes a = a \otimes 0 = 0$.

Semirings can be used to combine two matrices by generalising the matrix product.

Definition 4 (Matrix product based on semirings). The matrix product for two matrices $\mathbf{C}$ and $\mathbf{F}$ based on a semiring $(\sigma, \oplus, \otimes)$ is defined as follows:

$$(\mathbf{C} \odot \mathbf{F})_{r,c} = \oplus_i (\mathbf{C}_{r,i} \otimes \mathbf{F}_{i,c}).$$

The traditional matrix product is the matrix product based on the $(\mathbb{R}, +, \times)$ semiring. As shown earlier, we cannot use this semiring as in the resulting matrix the resulting values may not be valid.

Hence, we require a different semiring. In this article, we will mainly focus on the *max-product* semiring, even though other choices are also possible, such as the *min-product* semiring.

Definition 5 (Max-product semiring). The max-product semiring is the semiring $(\sigma, \oplus, \otimes)$ in which $a \oplus b = \max(a, b)$ and $a \otimes b = a \times b$.

Example 5 (Max-product semiring).

$$\begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 1 \\ 0 & 1 \\ 0 & 1 \end{pmatrix} \odot \begin{pmatrix} 1 & 2 & 5 & 6 & 0 & 0 \\ 0 & 0 & 4 & 6 & 1 & 2 \end{pmatrix} \equiv \begin{pmatrix} 1 & 2 & 5 & 6 & 0 & 0 \\ 1 & 2 & 5 & 6 & 1 & 2 \\ 1 & 2 & 5 & 6 & 1 & 2 \\ 0 & 0 & 4 & 6 & 1 & 2 \\ 0 & 0 & 4 & 6 & 1 & 2 \end{pmatrix}$$

Using the max-product semiring, we can combine the two factorised matrices in Example 4 into a single matrix; the two partial rank profiles are aggregated for user 2 and 3 by taking the maximum (green values).

Note that the max-product semiring chooses the highest rank in case two ranks overlap. With a min-product semiring the lowest value would be chosen.

Another important choice in the rank matrix factorisation framework is the choice of the scoring function f. In this article we will limit our attention to *additive* scoring functions.

Definition 6 (Additive scoring function). Given two matrices $\mathbf{M}$ and $\mathbf{R}$, a scoring function $f(\mathbf{M}, \mathbf{R})$ is additive if we can write the scoring function as follows:

$$f(\mathbf{M}, \mathbf{R}) = \sum_{r=1}^{m} \sum_{c=1}^{n} \delta(\mathbf{M}_{r,c}, \mathbf{R}_{r,c}),$$

where $\delta : \sigma \times \sigma \to \mathbb{R}$ scores the difference between values $\mathbf{M}_{r,c}$ and $\mathbf{R}_{r,c}$.

The main arguments in favour of these choices are that they are conceptually easy and that they enable more efficient algorithms.

In subsequent sections we will demonstrate how to apply this framework on two rank data mining problems, namely Sparse RMF [7] and rank matrix tiling [1].

3 Sparse mRMF

In this section, we study the first problem instance of sRMF, called *Sparse Max-product Semiring Rank Matrix Factorisation* or *Sparse mRMF*, which aims to find patterns of consistent ranks, such as illustrated in Example 1. This problem was first introduced in our previous work [7] and named *Sparse RMF* there. Another example of Sparse mRMF is provided below.

Example 6 (Sparse mRMF example).

$$\mathbf{M} = \begin{pmatrix} 1 & 2 & 5 & 4 & 6 & 3 \\ 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 3 & 2 & 4 & 5 & 6 \\ 2 & 3 & 5 & 6 & 4 & 1 \\ 2 & 3 & 5 & 6 & 1 & 4 \end{pmatrix} \mathbf{C} = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix} \mathbf{F} = \begin{pmatrix} 1 & 2 & 0 & 4 & 0 & 0 \\ 1 & 0 & 0 & 4 & 5 & 6 \\ 2 & 3 & 5 & 6 & 0 & 0 \end{pmatrix}$$

This example shows a rank matrix approximated by the product of two smaller matrices. Rank matrix $\mathbf{M}$ consists of five rows and six columns. Assuming no ties and complete rankings, each row contains each of the numbers 1 to 6 exactly once. Sparse mRMF factorises matrix $\mathbf{M}$ into the product of a binary 5×3 matrix named $\mathbf{C}$ and a 3×6 rank matrix named $\mathbf{F}$. Each row of matrix $\mathbf{F}$ is a sparse rank vector with many zeros and can be interpreted as a local pattern.

Let $\mathbf{R}$ be the reconstructed matrix of the factorisation, i.e., $\mathbf{R} = \mathbf{C} \odot \mathbf{F}$. We say an entry $\mathbf{R}_{r,c}$ is *covered* if $\mathbf{R}_{r,c} > 0$. We define *coverage* of a factorisation as follows:

$$\mathit{coverage}(\mathbf{R}) = \sum_{\mathbf{R}_{r,c} > 0} 1 \tag{2}$$

To support the aim of mining sparse patterns in rank matrices, the scoring function δ (Definition 6), which measures the similarity between matrix $\mathbf{M}$ and matrix $\mathbf{R}$, needs to be designed such that it: 1) rewards patterns that have a high coverage (ideally, the whole data would be covered), and 2) penalises patterns that make a large error within the cover of the factorisation. To achieve that aim, we define the scoring function δ as follows:

$$\delta(a, b) = \begin{cases} 0 & \text{if } b = 0; \\ \alpha - |a - b| & \text{otherwise.} \end{cases} \tag{3}$$

Here, a is an entry in matrix $\mathbf{M}$ and b is the corresponding entry in matrix $\mathbf{R}$. The term α defines how much reward is given for covering a entry in the data; the larger α is, the larger the patterns will be. The reward is lowered by penalizing for errors; for errors higher than α the term $\delta(a, b)$ will be negative. Hence, setting α low enough will ensure that we will not cover the complete data.

The error term $|a - b|$ is related to the *Footrule distance*, which is a well-known distance for comparing rankings.

Definition 7 (Footrule distance). Given two rank vectors, $\mathbf{u} = (u_1, \ldots, u_n)$ and $\mathbf{v} = (v_1, \ldots, v_n)$, the Footrule distance is defined as $\sum_{i=1}^{n} |u_i - v_i|$.

The α parameter balances errors against coverage. Indeed, an alternative way of writing our scoring function is:

$$f(\mathbf{M}, \mathbf{R}) = \alpha \cdot \mathit{coverage}(\mathbf{R}) - \mathit{error}(\mathbf{M}, \mathbf{R}),$$

where

$$\mathit{error}(\mathbf{M}, \mathbf{R}) = \textstyle\sum_{\mathbf{R}_{r,c} > 0} |\mathbf{M}_{r,c} - \mathbf{R}_{r,c}|.$$

Many other scoring functions could also be used to measure the disagreement between rows, for instance, Kendall's tau; see [19] for a survey. We choose the Footrule as it can be calculated relatively efficiently.

Note that we do not take into account the error for entries that are not covered; this reflects our interest in discovering *local patterns* that not necessarily characterise the complete data.

Plugging this scoring function in our earlier framework, we can summarise the problem of *Sparse mRMF* as follows.

Problem 1 (Sparse mRMF). Sparse mRMF is the rank matrix factorisation problem obtained by using

- the max-product semiring;
- the set of permissible values $\sigma_p = \sigma$;
- the additive scoring function based on formula (3).

4 Max-product semiring rank matrix tiling

We study the second instance of sRMF called *Max-product Semiring Rank Matrix Tiling* or *mRMT*. This problem was first introduced in our previous work [1] and named ranked tiling there. It aims to identify subsets of the data with high ranks, as illustrated in Example 2. Another example of mRMT is provided below.

Example 7 (mRMT example).

$$\mathbf{M} = \begin{pmatrix} 1 & 2 & 5 & 4 & 6 & 3 \\ 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 3 & 2 & 4 & 5 & 6 \\ 2 & 3 & 5 & 6 & 4 & 1 \\ 2 & 3 & 5 & 6 & 1 & 4 \end{pmatrix} \mathbf{C} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix} \mathbf{F} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 \end{pmatrix}$$

This example shows a rank matrix in which the regions with a rank higher than 3 are indicated by means of two Boolean matrices; the matrix **C** identifies the rows included in the tiles; the matrix **F** indicates the columns.

In comparison with Sparse RMF, in mRMT we do not require that the ranks of the items are (approximately) the same between different rows included in the tile; we only require that sufficiently high ranks are included in a tile.

To formalise the mRMT problem, our first choice is to limit the set of permissible values to $\{0, 1\}$; as a result, we only characterise the columns included in the tiles.

Next, we define the scoring function δ as follows:

$$\delta(a, b) = \begin{cases} 0 & \text{if } b = 0; \\ a - \theta & \text{otherwise.} \end{cases} \tag{4}$$

Again, in this scoring function we only look at those entries of the rank matrix covered by the tiles. Here, however, we give a higher score for a covered entry if its rank is higher; if the rank is too low, the contribution of the entry may be negative, which will discourage covering too many entries with low ranks[1].

Note that the effect of the parameter θ is the opposite of the parameter α in Sparse mRMF: the higher we choose the value θ, the smaller will be the tiles that will be found; the higher we choose the value α, the larger the factors we will find.

In summary, the mRMT problem can be defined as follows.

Problem 2 (mRMT problem). The mRMT problem is the rank matrix factorisation problem obtained using

- the max-product semiring;
- the set of permissible values $\sigma_p = \{0, 1\}$;
- the additive scoring function based on formula (4).

Note that the covered regions obtained by mRMT and Sparse mRMF do not need to overlap. Sparse mRMF obtains a decomposition with a low error in a large region of the matrix. This covered region may consist of a part of the matrix with neither low nor high ranks, while mRMT focuses primarily on high ranks. In practice, however, there can be overlap between the results of the methods: for instance, if Sparse mRMF were configured to accept high noise and mRMT were configured to accept relatively low ranks, the region covered by mRMT is likely to be part of the region covered by Sparse mRMF as well. In particular, if $a - \theta > 0$ for a given entry in a tile, there is also a factorisation in which $\alpha - |b - a| > 0$ for the same entry; for instance, when one sets α to a value $\alpha > (a_{max} - \theta)$, where a_{max} is the highest covered rank in the tile.

5 Algorithm

In this section, we will demonstrate how semiring rank matrix factorisation problems can be solved. We will first present a generic algorithm; subsequently we will discuss the details for the two specific rank factorisation settings and the optimisations we use to solve the semiring rank factorisation problems more efficiently.

5.1 Generic Algorithm

First, we observe that semiring rank matrix factorisation is related to many well-known hard data mining problems, such as Boolean matrix factorisation [20] and tiling [21]. Exact algorithms are only likely to solve small instances. As we wish to be able to analyse larger data matrices as well, we will use a heuristic approach in this article. The algorithm is summarised in Algorithm 1. The algorithm is an EM-style algorithm, in which the matrix **F** is optimised given matrix **C**, and matrix **C** is optimised given matrix **F**, and we repeat the iterative optimisation until the optimal score cannot be improved any more. This strategy was used in our previous work [7] and in the context of matrix factorisation before (see [13]).

We need to initialise the iterative process in a reasonable way. The solution we choose is to initialise the matrix **C** using the well-known k-means algorithm. To compute the similarities of rank vectors in k-means, we use the Footrule scoring function. The k-means algorithm clusters the rows in k groups, which can be used to initialise the k columns of **C**. Note that this results in initially disjoint patterns, in terms of their covers, but the iterative optimisation approach may introduce overlap.

The remaining question is how to solve the two optimisation problems in the iterative loop. Our general approach is to formulate these problems as *integer linear programming* (ILP) problems. We will show next how this can be done in a generic manner.

1. Note that our choice for scoring the entries covered by tiles is slightly different from the one in our earlier work [1], where we used a more complex scoring function that also included a term that reflects the number of tiles that covers a entry in the data. While this choice was justified by the algorithm used in our earlier work, we will show that the new algorithm presented in this article does not require the use of this more complicated scoring function.

Algorithm 1 *sRMF algorithm*

Require: Rank matrix $\mathbf{M}$, integer k
Ensure: Factorisation $\mathbf{C}$, $\mathbf{F}$
1: Initialise $\mathbf{C}$ using k-means clustering
2: **while** not converged **do**
3: $\mathbf{F} \leftarrow$ Optimise equation (1) given $\mathbf{C}$
4: $\mathbf{C} \leftarrow$ Optimise equation (1) given $\mathbf{F}$
5: **end while**

5.2 Solving Sparse mRMF using Integer Programming

We will illustrate how to model semiring rank matrix factorisation problems by using Sparse mRMF as an example.

Theorem 1 (Optimisation model for Sparse mRMF). Solutions to the following optimisation model are solutions to the Sparse mRMF problem:

$$\text{maximise} \sum_i \sum_j (\alpha \mathbf{A}_{i,j} - \mathbf{Y}_{i,j}) \tag{5}$$

subject to

$$0 \leq \mathbf{C}_{i,t} \leq 1 \tag{6}$$
$$0 \leq \mathbf{F}_{i,t} \leq n \tag{7}$$
$$\mathbf{R}_{i,j} \geq \mathbf{C}_{i,t}\mathbf{F}_{t,j} \tag{8}$$
$$\mathbf{R}_{i,j} \leq \mathbf{C}_{i,t}\mathbf{F}_{t,j} + (1 - \mathbf{B}_{i,j,t})n \tag{9}$$
$$\mathbf{B}_{i,j,t} \in \{0,1\} \tag{10}$$
$$\sum_t \mathbf{B}_{i,j,t} = 1 \tag{11}$$
$$\mathbf{A}_{i,j} \in \{0,1\} \tag{12}$$
$$n\mathbf{A}_{i,j} \geq R_{i,j} \tag{13}$$
$$\mathbf{A}_{i,j} \leq R_{i,j} \tag{14}$$
$$\mathbf{M}_{i,j}\mathbf{A}_{i,j} - \mathbf{R}_{i,j} \leq \mathbf{Y}_{i,j} \tag{15}$$
$$-\mathbf{M}_{i,j}\mathbf{A}_{i,j} + \mathbf{R}_{i,j} \leq \mathbf{Y}_{i,j} \tag{16}$$

Here, $1 \leq i \leq m$, $1 \leq j \leq n$, $1 \leq t \leq k$. $\mathbf{M}$ is the given data matrix; variables $\mathbf{R}$, $\mathbf{C}$, $\mathbf{F}$, $\mathbf{B}$, $\mathbf{A}$ and $\mathbf{Y}$ are to be found.

The correctness of this model follows from the following arguments:

- variables $\mathbf{R}_{i,j}$ encode the result of $\mathbf{C} \odot \mathbf{F}$: formula (8) ensures that $\mathbf{R}_{i,j} \geq \max_t \mathbf{C}_{i,t}\mathbf{F}_{t,j}$; formulas (9)-(11) ensure that $\mathbf{R}_{i,j} \leq \mathbf{C}_{i,t}\mathbf{F}_{t,j}$ for one choice for t (indicated by $\mathbf{B}_{i,j,t}$), which in combination with formula (8) means that the maximum is chosen;
- formulas (12)-(14) ensure that variables $\mathbf{A}_{i,j}$ encode those entries that are covered by the factorisation; i.e., $\mathbf{A}_{i,j} = 1$ iff $\mathbf{R}_{i,j} > 0$;
- formulas (5), (15) and (16) encode the additive scoring function based on formula (3); formula (15) and (16) ensure that $\mathbf{Y}_{i,j} \geq |\mathbf{M}_{i,j}\mathbf{A}_{i,j} - \mathbf{R}_{i,j}| = |\mathbf{M}_{i,j}\mathbf{A}_{i,j} - \mathbf{R}_{i,j}\mathbf{A}_{i,j}| = |\mathbf{M}_{i,j} - \mathbf{R}_{i,j}|\mathbf{A}_{i,j}$; as we look for maximal solutions in formula (5), which can only be obtained when $\mathbf{Y}_{i,j}$ is minimal, we can conclude that $\mathbf{Y}_{i,j} = |\mathbf{M}_{i,j} - \mathbf{R}_{i,j}|\mathbf{A}_{i,j}$. The optimisation criterion for one entry (i,j) can then be written as $\alpha\mathbf{A}_{i,j} - |\mathbf{M}_{i,j} - \mathbf{R}_{i,j}|\mathbf{A}_{i,j} = (\alpha - |\mathbf{M}_{i,j} - \mathbf{R}_{i,j}|)\mathbf{A}_{i,j}$, which corresponds to equation (3).

Note that this modeling approach can trivially be modified to solve variations of the Sparse mRMF problem; e.g., to deal with a min-product semiring we only need to modify equations (8)–(11).

A problem with the model above is that it is not a linear model if we would need to search for both $\mathbf{F}$ and $\mathbf{C}$; equations (8) and (9) calculate a product over two matrices. However, if we assume that one of these is fixed, the model is linear, and consequently, in each iteration of our algorithm we can use *integer linear programming solvers*, which are specialised solvers for finding solutions to linear models such as the model above.

5.3 Solving mRMT using Integer Programming

A similar approach can be used for mRMT. The most straightforward model is a modification of the Sparse mRMF model, in which:

- formula 7 is modified to $0 \leq \mathbf{F}_{i,t} \leq 1$, to represent the different permissible values;
- formula 5 is modified into $\mathbf{A}_{i,j}(\mathbf{M}_{i,j} - \theta)$ to reflect the different optimisation criterion;
- formulas 15 and 16 are removed.

However, this model is unnecessarily complex. In mRMT we have $\mathbf{A}_{i,j} = \mathbf{R}_{i,j}$, while $\mathbf{A}_{i,j}$ can be calculated more efficiently. Instead, we can also use the following model.

Theorem 2 (IP model for mRMT). Solutions to the following optimisation model are solutions to the mRMT problem:

$$\text{maximise} \sum_i \sum_j (\mathbf{M}_{i,j} - \theta)\mathbf{A}_{i,j} \tag{17}$$

subject to

$$0 \leq \mathbf{A}_{i,j}, \mathbf{C}_{i,t}, \mathbf{F}_{t,j} \leq 1 \tag{18}$$
$$\mathbf{A}_{i,j} \leq \sum_t \mathbf{C}_{i,t}\mathbf{F}_{t,j} \tag{19}$$
$$n\mathbf{A}_{i,j} \geq \sum_t \mathbf{C}_{i,t}\mathbf{F}_{t,j} \tag{20}$$

Here, $1 \leq i \leq m$, $1 \leq j \leq n$, $1 \leq t \leq k$. $\mathbf{M}$ is the given data matrix; variables $\mathbf{A}$, $\mathbf{C}$ and $\mathbf{F}$ are to be found.

This model can be solved more efficiently as it contains a much smaller number of variables.

5.4 Efficient Parallel Search

In the solution discussed above, we repeatedly solve integer linear programs to determine $\mathbf{C}$ and $\mathbf{F}$. These integer linear programs are still hard to solve and involve finding $m \times k$ and $k \times n$ assignments, respectively, for matrices $\mathbf{C}$ and $\mathbf{F}$.

We use the following properties to make solving these integer linear programs more efficient:

- the reconstructed matrix $\mathbf{R}$ is calculated based on a matrix product over a semiring;
- the scoring function is additive.

The consequence of these properties is that for a given $\mathbf{C}$, optimal values for all columns of $\mathbf{F}$ can be determined independently of each other; similarly, for a given $\mathbf{F}$, the rows of $\mathbf{C}$ can be determined independently of each other.

Consider the case of determining $\mathbf{C}$ given $\mathbf{F}$, i.e., determining the optimal occurrences of given rank patterns for each row. Given that our scoring function is additive, the

error score for one row in the reconstructed matrix **R** will not affect the error made for another row. Furthermore, given the use of products based on semirings, a row in the reconstructed matrix **R** is only determined by the corresponding row in the matrix **C** and the complete rank matrix **F**.

We exploit this property by running the solver for each row of **C** to determine the optimal solution for each row independently.

A further consequence of the independence of rows is that we can determine row assignments in parallel to each other. Consequently, we can distribute the optimisation problem over multiple cores of a CPU.

To implement our system, we relied on the OscaR system [22], which is an open source Scala toolkit for solving Operations Research problems. OscaR supports a modelling language for ILP. We configured OscaR to use the Gurobi[2] IP solver as the back-end solver. A benefit of OscaR/Scala is that it has built-in support for exploiting multiple cores to solve independent optimisation problems in parallel.

6 Experiments with synthetic data

We experiment on two sets of synthetic datasets to 1) evaluate the algorithms and 2) compare the patterns that mRMT and Sparse mRMF identify. The first set of data was generated for our previous work [1], where it was mainly used to demonstrate the capabilities of mRMT, including its suitability for data having incomparable rows. We will use the second set of data to demonstrate that Sparse mRMF is capable of recovering order-preserving patterns.

6.1 Synthetic data with implanted tiles

For the first set of experiments we used synthetic data with incomparable rows, i.e., rows having different scales, to show that mRMT finds the relevant patterns in such data while bi-clustering methods do not. Since bi-clustering methods work on numeric data, we use a simple generative model to generate continuous data. This numeric data is then transformed to a rank matrix to apply mRMT. For bi-clustering, we choose the constant-row setting, as there are many bi-clustering algorithms designed for this type of pattern and it is conceptually close to mRMT.

Data generation [1]. To generate synthetic datasets, we first generate background data, and then implant a number of constant-row bi-clusters with higher average values.

The values within each row are sampled, with a certain probability, from one of two distributions: one that represents background noise and one that is likely to interfere with the implanted patterns. First, for each row r, we uniformly sample μ_r^1, μ_r^2 from two ranges:

$$\mu_r^1 \sim U(0,3), \forall r \in \mathcal{R} \tag{21}$$

$$\mu_r^2 \sim U(3,5), \forall r \in \mathcal{R} \tag{22}$$

Second, for every entry in a row, indicated by row r and column c, we sample a latent binary variable $X_{r,c}$ from a Bernoulli distribution $Bin(p, 1-p)$, given some p. Depending on the value of this latent variable, the data is sampled from either the low-average or high-average distribution:

$$\mathcal{D}_{r,c} \sim \begin{cases} N(\mu_r^1, 1) & \text{if } X_{r,c} = 1 \\ N(\mu_r^2, 1) & \text{otherwise} \end{cases} \tag{23}$$

To plant a constant-row bi-cluster in a submatrix $\mathcal{D}_{R,C}$, specified by R and C, we use the following two equations:

$$\forall r \in R, \quad \mu_r \sim U(3,5) \tag{24}$$

$$\forall r \in R, \quad \mathcal{D}_{r,c} \sim N(\mu_r, 1) \tag{25}$$

Equation 24 is used to sample a mean for every row in a bi-cluster. This mean is uniformly sampled from the range $[3 \ldots 5]$, which is higher than the sampling range used for the background ($[0 \ldots 3]$).

Using this procedure we generated seven 1000 rows $\times$ 100 columns datasets, one for each $p \in \{0.10, 0.15, 0.20, 0.25, 0.30, 0.35, 0.40\}$. We implanted five ranked tiles in each dataset. Figure 1a depicts the numerical dataset for $p = 10\%$, Figure 1b depicts its corresponding rank matrix.

Accuracy of the tiles found by mRMT. We now evaluate the ability of the algorithm to recover the implanted ranked tiles. We do this by measuring *recall* and *precision*, using the implanted tiles as ground truth. Overall performance is quantified by the *F1* measure, which is the average of the two scores.

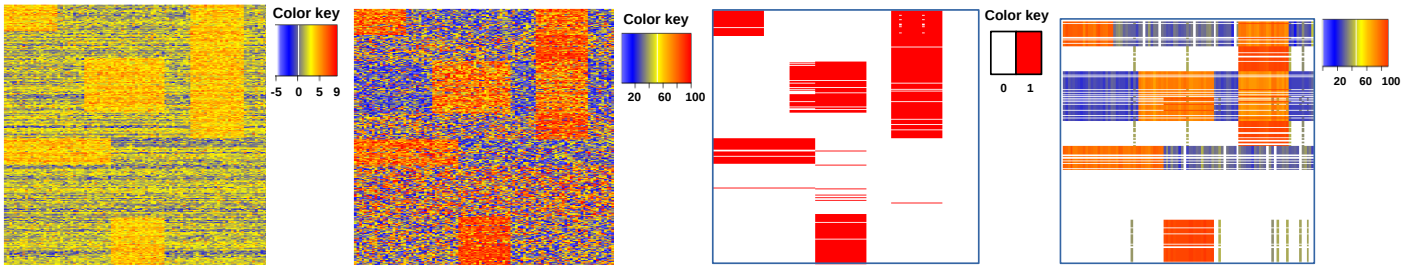
We varied the threshold θ and ran mRMT to factorise the rank matrix ($k = 5$) 10 times for each combination of θ and dataset. Then, the result that had the highest score was used to calculate the average precision, recall and F1 score over these combinations. Figure 2a summarises the results. When θ is around 60%, the algorithm achieves high accuracy (average F1 = 88%). At lower thresholds precision is low, while higher thresholds result in lower recall. This matches our expectation, as higher thresholds result in smaller tiles with higher values; this is also shown in Figure 5a, where the higher thresholds result in lower coverage (thus lower recall).

Comparison to cpRMT. In our previous work [1], we used *Constraint Programming for Rank Matrix Tiling* (*cpRMT*). cpRMT employs a greedy algorithm, i.e., it finds one tile, removes that tile and finds another one. Table 1 shows that the results obtained by mRMT are comparable to cpRMT. This demonstrates that the new approach behaves properly.

Comparison to Sparse mRMF. To contrast the two types of patterns that both methods discover, we also ran Sparse mRMF on the generated datasets. Figure 1d shows the reconstructed matrix produced by Sparse mRMF ($k = 5, \alpha = 20\%$) on the rank matrix generated with $p = 0.2$. It can be seen that the result produced by Sparse mRMF includes regions having low rank values (indicated in blue) instead of only focusing on the regions having high ranks (in red). As a result, its recall and precision are relatively low, which can also be seen in Figure 2b. This confirms that the two algorithms discover different types of rank patterns.

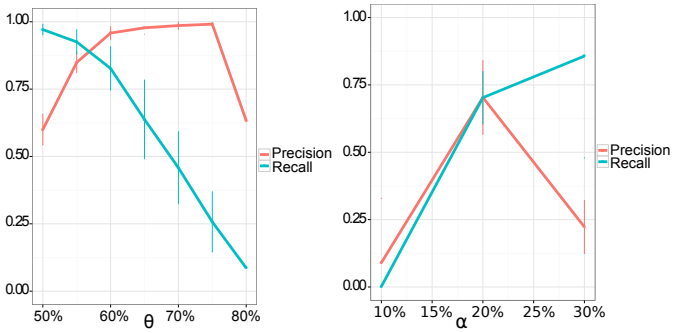
Comparison to Sparse pRMF. We ran the previous implementation of Sparse RMF [7], which uses linear algebra, i.e., the plus-product semiring, to calculate the matrix product,

2. http://www.gurobi.com/



(a) Numerical data (1000 × 100) (b) Rank matrix (1000 × 100) (c) mRMT (d) Sparse mRMF

Fig. 1: Recovering five implanted ranked tiles from the first set of synthetic data. Figure 1c shows the part of the matrix covered by mRMT ($k = 5, \theta = 60\%$). Figure 1d shows the reconstructed matrix obtained by Sparse mRMF ($k = 5, \alpha = 20\%$).



(a) mRMT (b) Sparse mRMF

Fig. 2: Sensitivity analysis for mRMT and Sparse mRMF on the synthetic data with implanted tiles.

and hence is named *Sparse Plus-product Semiring Rank Matrix Factorisation* or *Sparse pRMF* in this paper. Its rank profiles were much denser than the ones by Sparse mRMF, which resulted in low precision and very high recall (Table 1).

Comparison to bi-clustering. In the next experiment, we compare our approach to several bi-clustering algorithms. SAMBA [23] was designed for coherent evolution bi-clusters, in which there is coherence of the signs of values, i.e., up or down. The other methods discover coherent-valued bi-clusters, of which a constant-row bi-clusters are a special case. CC [24], Spectral [25], and Plaid [26] are available in the R biclust[3] package. FABIA[4] [27] and SAMBA[5] were downloaded from their respective websites. ISA [28] is from the R isa2 package[6].

Since large noise levels make the recovery task hard for any algorithm, we use one of the previously generated datasets with average noise level, i.e., $p = 0.20$. We ran all algorithms on this dataset and took the first five tiles/bi-clusters they produced. For most of the benchmarked algorithms, we used their default parameter values. For CoreNode, we used $msr = 1.0$ and $overlap = 0.5$, as preliminary experiments showed that this combination produced the best result. For ISA, we applied its built-in normalisation method before running the algorithm.

The results in Table 1 show that our algorithm achieves much higher precision and recall than the bi-clustering methods, which were run on the original data. Note that Spectral method [25] did not return any result. This indicates that when the rows in a numerical matrix are incomparable, converting the data to a rank matrix and applying mRMT is a better solution than applying bi-clustering.

3. http://cran.r-project.org/web/packages/biclust/
4. http://www.bioinf.jku.at/software/fabia/fabia.html
5. http://acgt.cs.tau.ac.il/expander/
6. http://cran.r-project.org/web/packages/isa2/

TABLE 1: Comparison of mRMT, Sparse mRMF, and bi-clustering methods. Precision, recall and F1 quantify how accurately the methods recover the 5 implanted tiles. $k = 5$.

Algorithm	Data type	Pattern	Precision	Recall	F1
mRMT	Ranks	Ranked tile	**95%**	81%	**88%**
cpRMT [1]	Ranks	Ranked tile	88%	**83%**	86%
Sparse mRMF	Ranks	Sparse rank profile	70%	70%	70%
Sparse pRMF [7]	Ranks	Sparse rank profile	26%	100%	63%
CoreNode [29]	Numerical	Coherent values	43%	72%	58%
FABIA [27]	Numerical	Coherent values	40%	24%	32%
Plaid [26]	Numerical	Coherent values	90%	6%	48%
SAMBA [23]	Numerical	Coherent evolution	67%	3%	35%
ISA [28]	Numerical	Coherent values	64%	44%	54%
CC [24]	Numerical	Coherent values	35%	22%	29%
Spectral [25]	Numerical	Coherent values	-	-	-

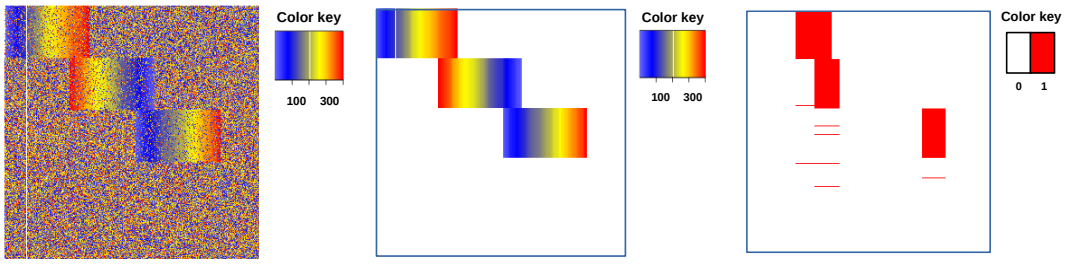
6.2 Synthetic data with implanted orders

In the second set of experiments, we evaluate the capability of Sparse mRMF and mRMT to recover consistent rankings of a subset of columns in a subset of rows. Hence, for this we generate rank data with implanted orders.

Data generation. For each dataset, we first implant three rank patterns and then generate its background information. Patterns are created by generating a reference rank profile and repeating this profile for a number of rows. Each reference rank profile is generated by uniformly sampling l integer numbers from the range $[1 \dots n]$, where l is the number of columns in the pattern. Noise is simulated by swapping w number of column pairs for each row in the pattern. After the patterns are implanted, each row is completed by a random permutation of the values in $[1 \dots n]$ not in its reference rank profile (i.e., the set difference).

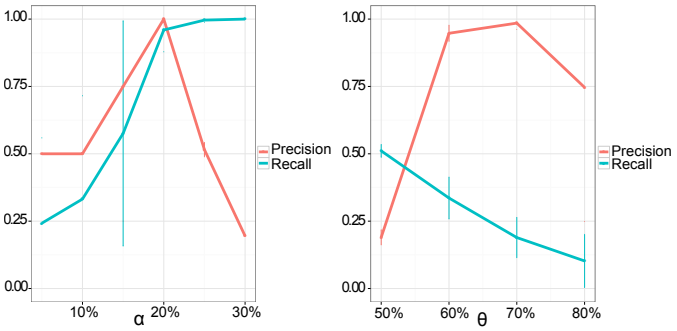
We generate four 1000 rows × 400 columns datasets, one for each $w \in \{0, 10, 20, 30\}$. In each dataset, we implant three overlapping order-preserving rank patterns, each of which spans 200 rows and 130 columns. Figure 3a shows the dataset generated with $w = 20$.

Accuracy of the recovered tiles by Sparse mRMF. We ran Sparse mRMF on the four simulated datasets with $k = 3$ and varying values for the threshold α, i.e., $\alpha \in \{10\%, 20\%, 30\%\}$. For each combination of threshold and



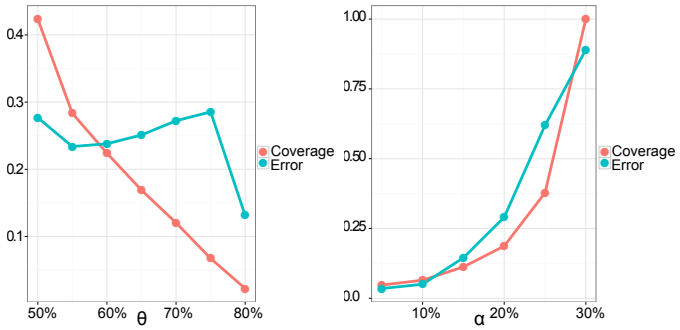
(a) Rank data (1000 × 400) (b) Sparse mRMF (c) mRMT

Fig. 3: Recovering three implanted rank profiles from the synthetic data with implanted orders. Figure 3a shows the data ($w = 20$). Figure 3b is obtained by Sparse mRMF with $k = 3, \alpha = 20\%$. Figure 3c is obtained by mRMT with $k = 3, \theta = 60\%$.



(a) Sparse mRMF (b) mRMT

Fig. 4: Sensitivity analysis for Sparse mRMF and mRMT on the synthetic data with implanted orders.



(a) mRMT (b) Sparse mRMF

Fig. 5: Evaluation of coverage and error scores for mRMT and Sparse mRMF on the synthetic data with implanted tiles and respectively on the synthetic data with implanted orders.

dataset, we ran the algorithm 10 times and used the result that had the highest score. Similar to the previous experiments, we also used precision and recall to evaluate recovery accuracy. Figure 4a shows that Sparse mRMF succeeds in recovering the three simulated patterns with high precision and recall when $\alpha = 20\%$. As expected, Sparse mRMF obtains high recall and low precision when the threshold is increased (to 30%); Figure 5b shows that both coverage and error increase steeply. When the threshold is (too) low, i.e., $\alpha = 10\%$ in this case, the implanted patterns cannot be recovered. Hence, for this case, Sparse mRMF has low value for both precision and recall.

Accuracy of the recovered tiles by mRMT. To contrast the patterns that Sparse mRMF and mRMT discover, we also run mRMT on this data. Figure 3c displays the regions covered by the best result produced by mRMT (on the synthetic data shown in Figure 3a and with threshold $\theta = 60\%$). It can be seen that these regions contain the high ranks of the implanted patterns. In other words, mRMT only partially recovers the reference rank profiles, which explains the low recall in Figure 4b. This again confirms our expectations, as Sparse mRMF and mRMT were designed for different types of rank patterns.

We did not run our previous implementation of Sparse RMF [7] on this dataset. This is because Sparse mRMF is a more general setting and its results are a natural choice to compare with mRMT. We also did not compare with BMF [20] and the traditional tiling method [21] as converting rank data to Boolean data results in information loss.

7 Real world case studies

In this section we report on three real world case studies concerning the European Song Festival, breast cancer subtypes, and sushi consumption.

7.1 European Song Festival dataset

The Eurovision Song Contest (ESC) has been held annually since 1956. Each participating country gives voting scores, which are a combination of televoting and jury voting, to competing countries. Scores are in the range of $1 \ldots 8$, 10, and 12. Each country awards 12 points to its most favourite country, 10 points to its second favourite, and $8 \ldots 1$ to the third $\ldots$ tenth favourites respectively. The data can be represented by a matrix in which rows correspond to voting countries, columns correspond to competing countries, and entry values to the scores.

The ESC dataset, collected and processed by Le Van et al. [1], consists of 44 rows and 37 columns, corresponding to 44 voting countries and 37 competing countries, for the final rounds of the period 2010 – 2013.

Running experiments. We ran Sparse mRMF with varying values for the threshold α, i.e., $\alpha \in \{5\%, 10\%, \ldots, 30\%\}$, for each of which we factorised the rank matrix with different k values, i.e., $k \in \{5, \ldots, 12\}$. Figure 6a shows the average coverage and error scores for the different α values. It can be seen that, for $\alpha = 5\%$, Sparse mRMF made almost no mistake (error = 0.04) but coverage is low (19%). When $\alpha = 30\%$, on the other hand, Sparse mRMF covers almost the entire matrix ($> 90\%$) while having an average entry-based error of 5, which might be acceptable as the range

of the rank score in this dataset is $[1 \dots 37]$. In practice, we would choose a threshold value based on coverage and/or error depending on the background knowledge and preferences of the data miner. Here we choose $\alpha = 10\%$ as the corresponding error is low and the coverage is substantially higher than that for $\alpha = 5\%$. Given α, we next have to decide an appropriate value for k. For this we examine Figure 6b, which shows the coverage for $\alpha = 10\%$ and varying k. We choose $k = 10$ as coverage appears to be stable for higher k.

For mRMT we use the same parameter selection procedure as for Sparse mRMF. Figure 6c depicts average coverage and error scores obtained by mRMT with varying θ values. We choose $\theta = 80\%$ as both coverage and error decrease slowly beyond that point. Similarly, we choose $k = 10$ based on Figure 6d. This configuration is also used for cpRMT.

Voting patterns. The heatmaps of the reconstructed matrix obtained by Sparse mRMF and the covered matrix by mRMT are shown in Figures 7b and 7c respectively. Compared to the original rank matrix in Figure 7a, the two heatmaps show that the two methods strongly *sparsified* the original rank matrix. The figures also show that the rank profiles produced by Sparse mRMF contain both high and low ranks while the ones produced by mRMT only indicate the places where high ranks appear, as expected.

Table 2 illustrates the benefits of using the new formalisation for rank matrix factorisation. The result shows that compared to Sparse pRMF [7], the old model, Sparse mRMF, the new model, attains higher coverage (32% compared to 30%) and a lower error (1.12 compared to 1.59). Sparse mRMF also enjoys a substantial increase of the overlap among the rank profiles in their covered rows. However, more computation is needed.

To show how the rank profiles produced by the two methods can provide insight into the data, we visualise two rank profiles of each method in Figure 8. They show the typical voting behaviour of Western European countries towards Nordic countries (Figures 8b and 8d), and that of Eastern European countries toward some other countries (Figures 8a and 8c). For example, countries in Eastern Europe tend to give higher scores to Russia and Nordic countries than to other countries. In general, the discovered patterns confirm that countries tend to give high scores to their neighbours, which confirms common knowledge about the European Song Contest.

7.2 Discovering breast cancer subtypes

Breast cancer is known to be a heterogeneous disease that can be categorised in clinical and molecular subtypes [30]. Assignment of patients to such subtypes is crucial to give adapted treatments to patients. Computational models have been proposed to integrate multiple data types and discover cancer subtypes [31], [32], but these integrative subtyping methods do not explicitly extract subtype-specific features. The goal of this case study is to demonstrate that: 1) we can integrate multiple data types that are inherently incomparable but can be compared when transformed to rank data; 2) we can simultaneously discover breast cancer subtypes and their subtype-specific features.

TABLE 2: Performance statistics of the sRMF algorithms on the European Song Festival dataset to discover $k = 10$ rank profiles. The *error* score is the average error in the covered area when the score is an absolute value, and is the percentage of entries in the covered region having ranks below the threshold θ when the score is a relative. The *sparsity* score is the average percentage of 0s in rank profiles. The *overlap* score is the percentage of the covered rows present in more than 1 rank profile.

Algorithm	Coverage	Error	Sparsity	Overlap	Time/run
Sparse pRMF [7]	30%	1.59	59.7%	2%	3s
Sparse mRMF	32%	1.12	52.4%	30%	69.2s
mRMT	9.4%	3.3%	96.7%	95.5%	0.36s
cpRMT [1]	10.5%	10%	94.6%	82%	20s

The case study we present here concerns a simplified setting of the one in our recent work [8]. We here consider a single, integrated rank matrix and focus on mRMT.

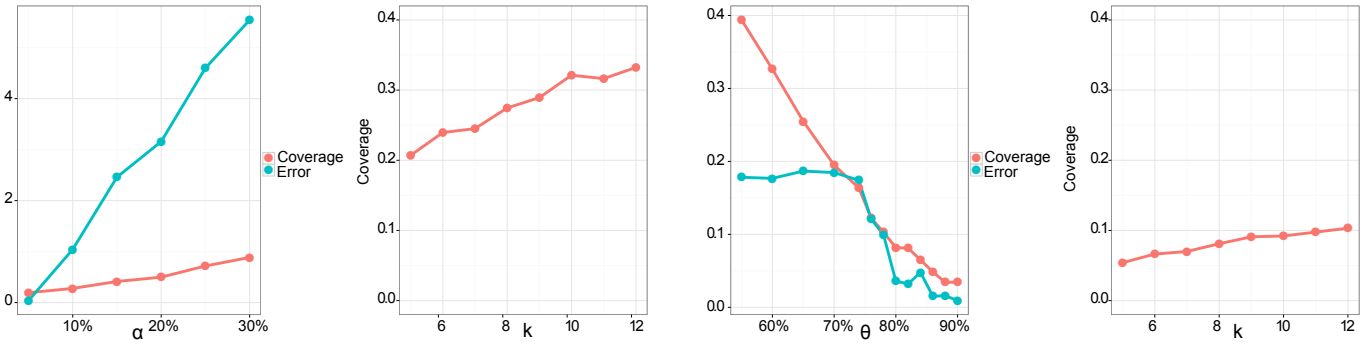
Data pre-processing. We use the well-studied TCGA breast cancer dataset [30], which provides the following four data types for the same set of samples (patients): mRNA measured by microarray technology, microRNA measured by RNA-Seq, proteins, and copy number variations (CNV).

We first selected all the tumour samples that have measurements at the four molecular levels, which resulted in 363 samples. Second, we filtered mRNA and microRNAs as in our previous study [1]. That is, we selected genes based on their differential expression relative to normal (non-tumour) samples. The filtering step resulted in 1761 mRNAs out of 17814 mRNAs and 138 microRNAs out of 1222 microRNAs. Third, we used all the protein data (131 proteins), which were post-processed by the UCSC genome browser [33]. Finally, copy number regions (82 in total) were identified with GISTIC tool [34], of which the analysis result was provided together with the TCGA paper [30]. Finally, each data level was converted to ranks and combined into a single rank matrix consisting of 2112 rows and 363 columns.

Running experiments. As it is our aim to discover cancer subtypes consisting of a number of tumor samples having consistently similar expression patterns, Sparse mRMF is not suitable for this type of application. Hence, for this case study we restrict ourselves to mRMT.

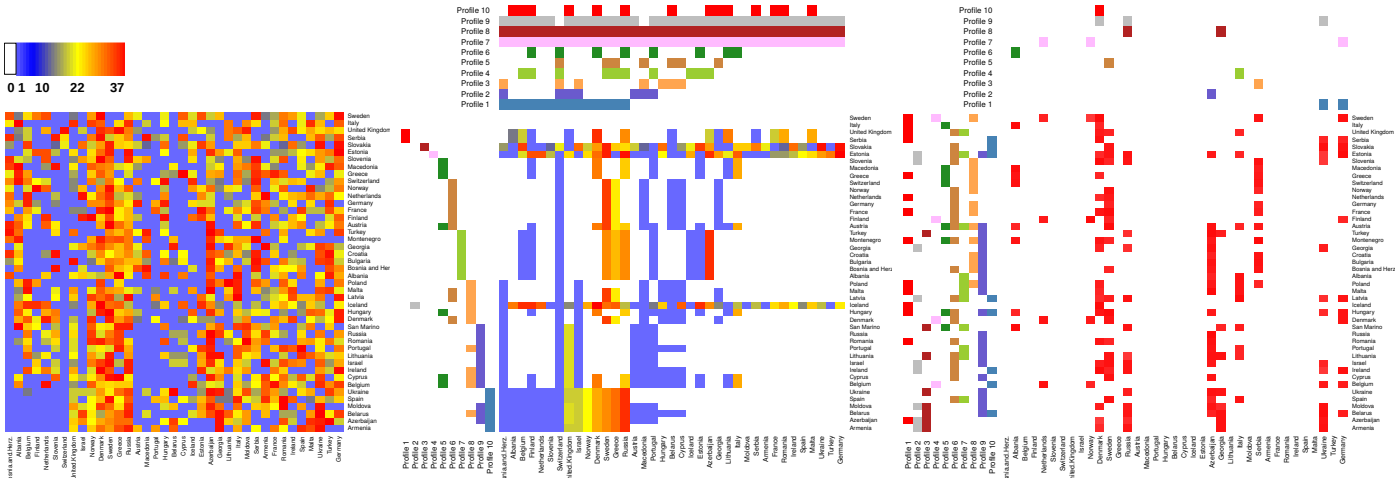
We ran the parallel implementation of the mRMT method on the TCGA breast cancer dataset with $\theta \in \{55\%, 60\%, \dots, 90\%\}$ and $k \in \{5, \dots, 16\}$. For each combination of the two parameters, we ran the algorithm 100 times and took the best result. Figure 9a shows the obtained *coverage* and *error* scores with varying θ, from which we can infer that there is no clear cut to choose θ in this case. In general, the higher the threshold value, the lower the error and the coverage. To trade off the coverage and the error, we chose $\theta = 65\%$. Given the selected θ, we next had to decide the value for k. We plotted the coverage score w.r.t k (Figure 9b) and then decided to stop at $k = 10$ as we found the coverage score to increase only very slowly after that point.

With $\theta = 65\%$ and $k = 10$, the average running time for one run is 432s on a desktop computer (Intel(R) Core(TM) i7-2600 CPU @ 3.40GHz, 8 threads, 16GB RAM). With the chosen parameter values, the algorithm produces 10 over-



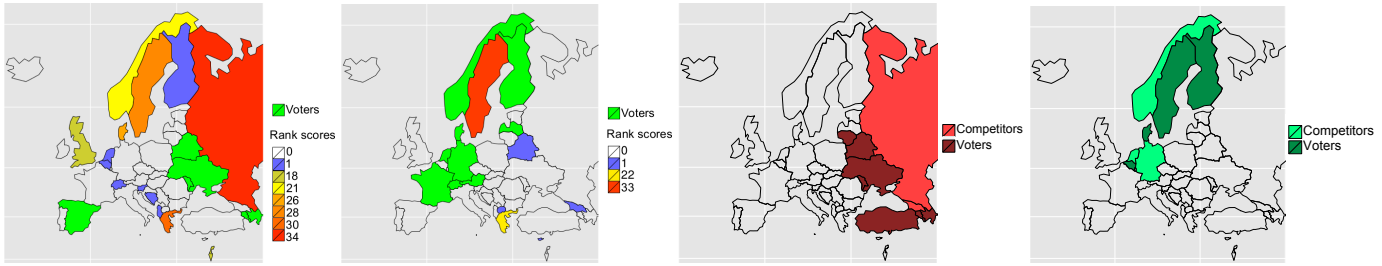
(a) Sparse mRMF (b) Sparse mRMF (c) mRMT (d) mRMT

Fig. 6: Parameter tuning for Sparse mRMF and mRMT on the European Song Festival dataset.



(a) Rank data (b) The reconstructed matrix by Sparse mRMF (c) The part of the data covered by mRMT

Fig. 7: EU Song Festival results show how Sparse mRMF and mRMT focus on specific structure and hence sparsify the data. Figure 7a and Figure 7b have the same color key. In Figure 7c, *"red"* is 1 and *"white"* is 0.



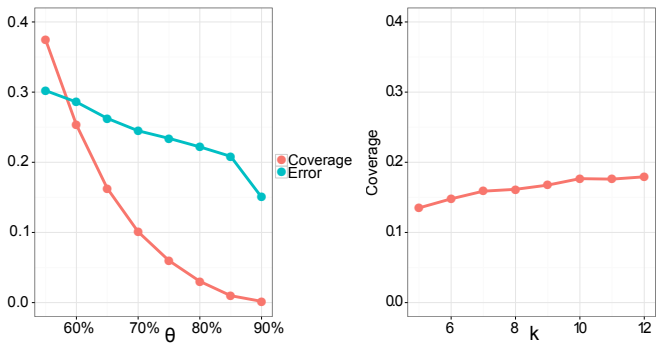
(a) Tile 1 (Sparse mRMF) (b) Tile 6 (Sparse mRMF) (c) Tile 8 (mRMT) (d) Tile 7 (mRMT)

Fig. 8: Rank patterns discovered on the ESC dataset by Sparse mRMF and mRMT. For the results by Sparse mRMF (Figure 8a and Figure 8b), the rank profiles, which depict the obtained voting scores of competitors, are painted in red; the corrresponding rows (voting countries), which show the places where these rank profiles appear, are painted in green. For the results by mRMT (Figure 8c and Figure 8d), voting countries are painted in dark colors and competing countries are painted in light colors.

lapping ranked tiles. Though the overlap structure can be useful to study the similarity among the discovered subtypes, for practical reasons we decided to choose a simple interpretation in which each sample is assigned to a single subtype. With this aim, we developed a post-processing step in which each sample belonging to multiple subtypes according to the mRMT method is assigned to the subtype giving the highest rank score in Equation (17). Figure 10 shows the result obtained using this procedure.

Subtype analysis. First, we observe that most discovered subtypes comprise all four types of features. The exceptions are subtypes S1, S3, S5 and S7, which have mRNA, miRNA and protein features but lack CNVs.

Next, we test to what extent the discovered subtypes agree with known clinical information. To this end, we use the PAM50 annotation [35], which classifies breast cancer patients into four subtypes, Luminal A, Luminal B, Basal and Her2, using the expression of 50 mRNAs. Figure 11 shows that our approach does not only match the PAM50 classification to a large extent, it also further refines known



(a) (b) $\theta = 65\%$

Fig. 9: Parameter tuning on breast cancer dataset.

subtypes. For example, our approach recovered the Basal subtype in subtype S4; recovered the Her2 subtype in subtype S9 and sub-divided the Lumial A subtype into four smaller groups, namely, subtype S2, S3, S5 and S10.

Comparison with cpRMT. We next compare mRMT with *cpRMT* [1]. For a fair comparison, we ran cpRMT 100 times, repeating large neighbourhood search (LNS) as many times as repeating mRMT procedure. Figure 12 shows that mRMT obtains higher coverage and lower error scores than cpRMT. This matches with our expectation as mRMT implementation employs a *global optimisation* procedure, whereas cpRMT uses a *greedy approach*, i.e., it finds one 'maximal' ranked tile, removes it, and proceeds to find the next one.

Overall, we conclude that mRMT can identify cancer subtypes and their features from several data types by searching for patterns in rank data.

7.3 The Sushi dataset

The Sushi dataset, collected by Kamishima [4], contains preferences of 5000 people over ten different sushi types.

We ran both Sparse mRMF and mRMT on this dataset. With Sparse mRMF, we chose the same threshold values ($\alpha = 20\%$; $k = 8$) as we used in our previous work [7] for a fair comparison. With mRMT and cpRMT [1], we used $\theta = 65\%$ to select subjectively high rank values and $k = 8$ as in Sparse mRMF.

Table 3 shows that the rank profiles found by Sparse mRMF have much larger overlaps and are sparser than the previous model Sparse pRMF [7], while the average error per covered entry is smaller. The only drawback are the longer runtimes that come with these improved results. cpRMT is slightly better than mRMT, as in this case the number of the columns of the matrix is quite small, i.e., 10.

Figure 13 shows the eight rank profiles found by Sparse mRMF and mRMT. In general, the customers have clear preferences over the ten sushi types. For example, the first rank profile F1 in Figure 13a depicts that there is a group of customers preferring light sushi types, such as maguro, ebi, and tekka maki, to oily and seasoning sushi like uni and anago. Interestingly, the eighth rank profile F8 maintains that there exists a group of customers who have completely the opposite taste, preferring anago sushi to ebi sushi. We observe similar rank patterns (but only for high ranks) in the mRMT results in Figure 13b.

TABLE 3: Performance statistics of the sRMF algorithms on the Sushi dataset [4] to discover $k = 8$ rank profiles. The scores have the same meaning as those in Table 2.

Algorithm	Coverage	Error	Sparsity	Overlap	Time/run
Sparse pRMF [7]	78.2%	1.31	13.8%	0%	53mi
Sparse mRMF	77.2%	1.22	41.3%	75.2%	2h32mi
mRMT	34.7%	7.3%	82.5%	92%	204s
cpRMT [1]	39%	3.6%	87.5%	99%	1h52mi

8 Related Work

Rank aggregation [36] studies the problem of finding a single rank profile that has the least discrepancy with all of the rankings over the items provided by the users in a database. This problem appears in domains such as aggregating user preferences [37] and combining search results [38]. Although rank aggregation is obviously related to the Sparse mRMF setting with $k = 1$, there are still major differences between the two lines of research. First, the rank profile produced by rank aggregation is typically *dense* as rank aggregation aims to find a profile that can match with the rankings of the users as much as possible. In contrast, our Sparse mRMF aims to find a *sparse* representation, which is a subset of the complete ranking over the items that has the smallest error of rank values in the largest number of users. Second, our work on rank matrix factorisation aims at a framework to mine different types of data regularities in rank data while rank aggregation only aims at one type of rank pattern.

Discovering clusters of rankings is also related to our work. A non-exhaustive list of work in this direction includes [3], [39], [40], [41], [42], [43]. Similar to the rank aggregation setting discussed above, such methods look for complete rankings of all items, which are typically dense. In contrast, our sRMF model aims to find multiple types of rank patterns, which are sparse. Besides that, these methods only aim at one type of rank pattern while our work aims at a framework for mining different rank patterns.

Traditional matrix factorisation methods aim to find factorisations that are as close as possible to the original matrix, while sRMF is in favor of factorisations that can capture patterns users are interested in. This means that we do not aim to completely recover the original matrix, as long as the detected patterns can capture the main structure in the matrix. In this regard, our work is related to *sparse dictionary learning* [16], [44], [45]. However, the strategy to define the optimisation functions to find the sparse encodings is different. In sparse dictionary learning [16], a distance function is defined to minimise the error between the original matrix and the reconstructed matrix for all data points; it encourages to cover less of the data by using regularisation. In Sparse mRMF and mRMT, the scoring functions are defined in such a way that covering more data is encouraged, while having lower error in the covered region rather than in the whole data matrix.

sRMF uses semirings [18] to define the matrix product to factorise rank matrices. Karaev et al. [46] also used this idea to factorise non-negative real-valued matrices. However, Karaev et al. [46] aimed at a factorisation of which the reconstructed matrix is as close to the original matrix as possible. Hence, the factorised matrices are typically dense,

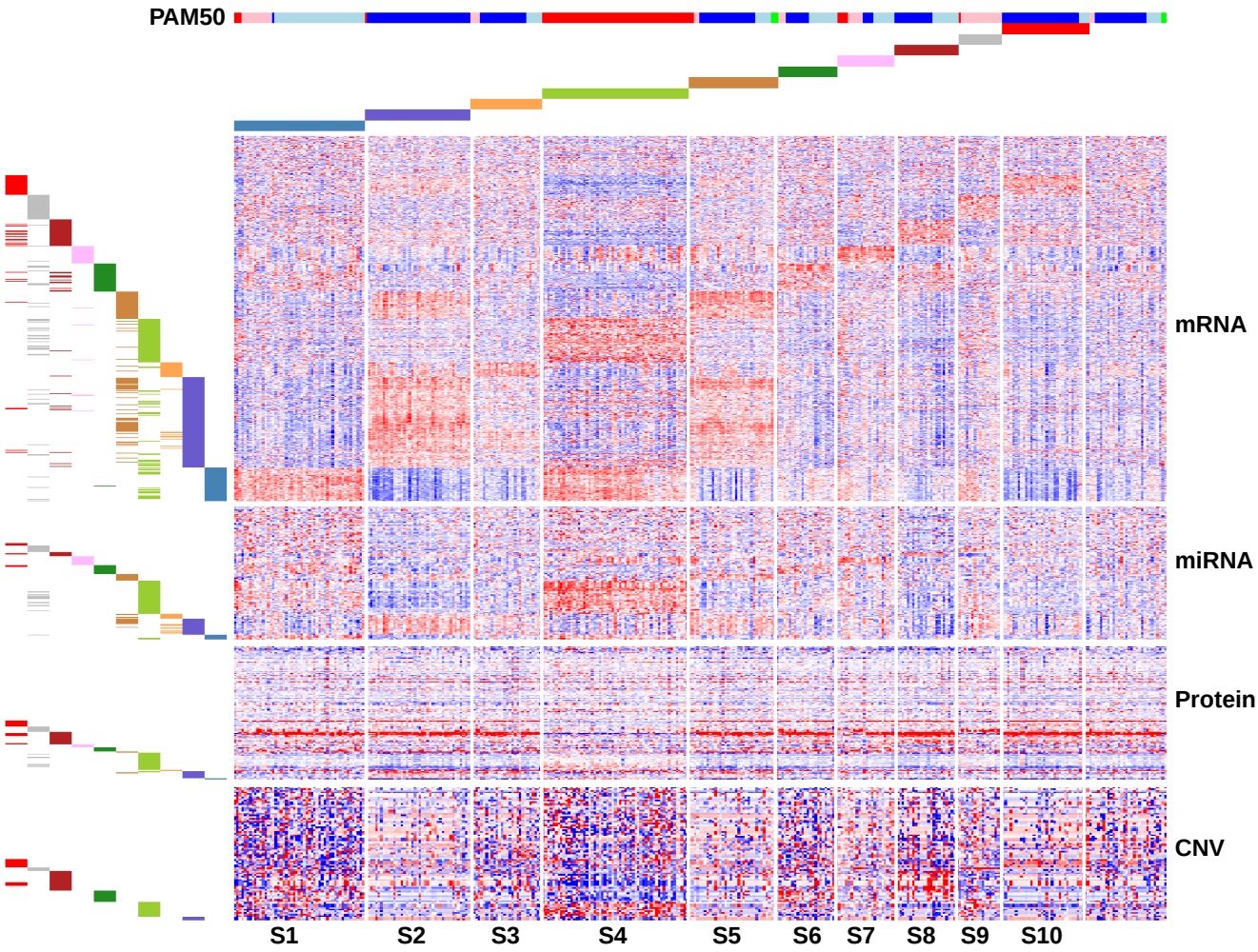


Fig. 10: mRMT on heterogeneous breast cancer data. The rows correspond to mRNA, miRNA, protein and CNV levels, the columns correspond to breast cancer samples. High expression values are represented by red, low expression by blue. The left and upper color bars indicate the tiles. PAM50 subtypes are indicated at the top (LuminalA=blue, LuminalB=light blue, Basal=red and HER2=pink).

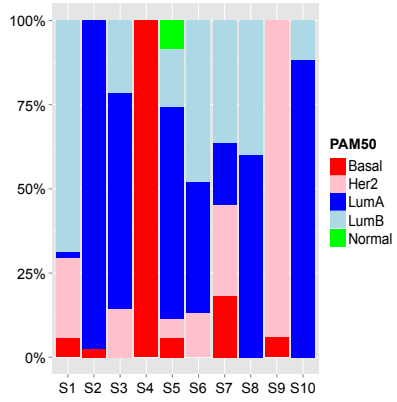


Fig. 11: Percentages of PAM50 samples in the subtypes discovered in the breast cancer data.

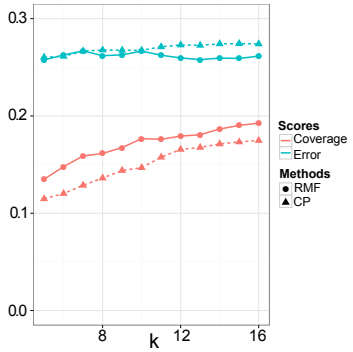
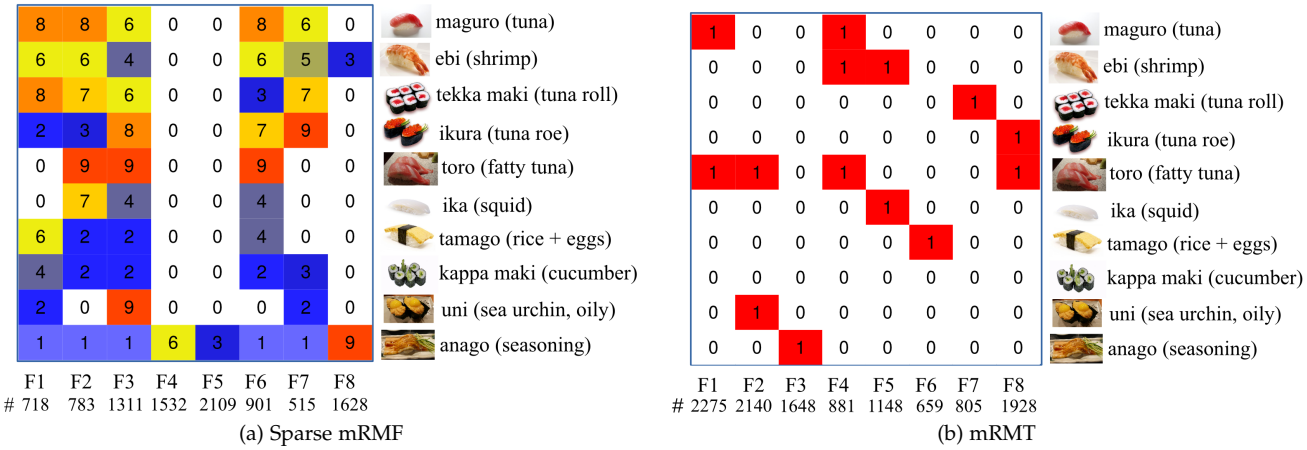


Fig. 12: Comparing mRMT with cpRMT on the breast cancer dataset. In both cases, θ was set to 65%.

which contrasts with our proposed models.

sRMF identifies sparse rank profiles that occur in a number of rows, which can be used to show local associations between subsets of columns and subset of rows. Hence, our work is related to bi-clustering [17] and tiling [21], but is different because of the type of regularities it aims to find. The literature describes four types of bi-clusters: constant-valued, constant-row, constant-column and coherent [17]. In



(a) Sparse mRMF (b) mRMT

Fig. 13: Sushi preference patterns found by Sparse mRMF and mRMT. The bottom row below the heatmaps, indicated by #, shows the number of users having the corresponding rank profile.

mRMT, the absolute values within the specified areas matter, i.e., the values must be higher than a given threshold. In Sparse mRMF, a ranking over a subset of items must be consistent in a number of rows. This is clearly different from the objectives of bi-clustering, as also demonstrated by the results presented in Section 6.1. In tiling, the task is to search for set of tiles (that is, a *tiling*) in a 0/1 matrix, which is obviously different from the rank data that we investigate in this paper. Note that BMF [20] can discover a set of *noisy* tiles to approximate Boolean matrices. By constraining $\sigma_p = \{0, 1\}$ and using an appropriate scoring function, sRMF can also perform BMF [20] and tiling [21].

9 Conclusions

Rank data is ubiquitous and useful. To discover regularities hidden in this type of data, new methods are needed.

We developed a generic semiring rank matrix factorisation framework for mining sets of patterns. We proposed to use a max-product semiring defined on permissible rank values of the data to calculate the matrix product of the two factorised matrices. To mine a specific type of data regularity, we proposed to use the two factorised matrices to define the patterns of interest by constraining the values of these matrices as well as an appropriate scoring function to measure the quality of the factorisation. We demonstrated how the proposed framework can be applied on two existing rank data mining problems, namely rank matrix tiling [1] and Sparse RMF [7]. Modelling the two problems using this framework illustrates the expressiveness and flexibility of the approach. Experiments on both synthetic datasets and real world problems show that the framework is capable of discovering different types of structure as well as obtaining high quality solutions.

In the future, we are interested in 1) developing model selection techniques for the selection of the value for hyper-parameter k, e.g., inspired by MDL-based techniques such as Krimp [10]; 2) applying the sRMF framework for other types of rank patterns; 3) improving the solving algorithms.

Acknowledgments

The authors would like to thank the support from KU Leuven [PF/10/010] (NATAR) and the Research Foundation–Flanders (FWO) project "Instant Interactive Data Exploration". T. Le Van was partially supported by a grant from CPER Nord-Pas de Calais/FEDER DATA Advanced data science and technologies 2015-2020. They also would like to thank Ana Carolina Fierro and Kathleen Marchal for their support.

References

[1] T. Le Van, M. van Leeuwen, S. Nijssen, A. C. Fierro, K. Marchal, and L. De Raedt, "Ranked Tiling," in *Proc. of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD-14) (2)*, 2014, pp. 98–113.

[2] P. Diaconis, *Group representations in probability and statistics*, ser. Lecture notes-monograph series. Institute of Mathematical Statistics, 1988.

[3] L. M. Busse, P. Orbanz, and J. M. Buhmann, "Cluster analysis of heterogeneous rank data," in *Proc. of the 24th international conference on Machine learning (ICML-07)*, 2007, pp. 113–120.

[4] T. Kamishima, "Nantonac collaborative filtering: Recommendation based on order responses," in *Proc. of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-03)*, 2003, pp. 583–588.

[5] Ben-Dor Amir, Chor Benny, Karp Richard, and Yakhini Zohar, "Discovering Local Structure in Gene Expression Data: The Order-Preserving Submatrix Problem," *Journal of Computational Biology*, vol. 10, no. 3-4, p. 373384, 2003.

[6] K. Deng, S. Han, K. J. Li, and J. S. Liu, "Bayesian Aggregation of Order-Based Rank Data," *Journal of the American Statistical Association*, vol. 109, no. 507, pp. 1023–1039, Oct. 2014.

[7] T. Le Van, M. van Leeuwen, S. Nijssen, and L. De Raedt, "Rank Matrix Factorisation," in *Proc. of the 19th Pacific-Asia conference on knowledge discovery and data mining (PAKDD-15)*. Springer, 2015, pp. 734–746.

[8] T. Le Van, M. van Leeuwen, A. C. Fierro, D. D. Maeyer, J. V. den Eynden, L. Verbeke, L. D. Raedt, K. Marchal, and S. Nijssen, "Simultaneous discovery of cancer subtypes and subtype features by molecular data integration," *Bioinformatics*, vol. 32, pp. i445–i454, 2016.

[9] S. Henzgen and E. Hüllermeier, "Mining rank data," in *Proc. of Discovery Science (DS-14)*. Springer, 2014, pp. 123–134.

[10] J. Vreeken, M. van Leeuwen, and A. Siebes, "Krimp: mining itemsets that compress," *Data Mining and Knowledge Discovery*, vol. 23, no. 1, pp. 169–214, 2011.

[11] D. Skillicorn, *Understanding Complex Datasets: Data Mining with Matrix Decompositions*. Chapman & Hall/CRC, 2007.

[12] L. Eldén, *Matrix Methods in Data Mining and Pattern Recognition*. Society for Industrial and Applied Mathematics, 2007.

[13] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorisation techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[14] J. P. Brunet, P. Tamayo, T. R. Golub, and J. P. Mesirov, "Metagenes and molecular pattern discovery using matrix factorization," *PNAS*, vol. 101, pp. 4164–4169, 2004.

[15] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, 1999.

[16] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online Learning for Matrix Factorization and Sparse Coding," *Journal of Machine Learning Research*, vol. 11, pp. 19–60, 2009.

[17] S. C. Madeira and A. L. Oliveira, "Biclustering algorithms for biological data analysis: a survey." *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 1, no. 1, pp. 24–45, 2004.

[18] J. S. Golan, *Semirings and Their Applications*. Kluwer, 1999.

[19] J. I. Marden, *Analyzing and Modeling Rank Data*. Chapman & Hall, 1995.

[20] P. Miettinen, T. Mielikainen, A. Gionis, G. Das, and H. Mannila, "The discrete basis problem," *IEEE Transactions on Knowledge and Data Engineering*, vol. 20, no. 10, pp. 1348–1362, 2008.

[21] F. Geerts, B. Goethals, and T. Mielikäinen, "Tiling Databases," in *Discovery Science (DS-04)*, 2004, pp. 278–289.

[22] OscaR Team, "OscaR: Scala in OR," 2012, available from `https://bitbucket.org/oscarlib/oscar`.

[23] A. Tanay, R. Sharan, and R. Shamir, "Discovering statistically significant biclusters in gene expression data," *Bioinformatics*, vol. 18, no. Suppl. 1, pp. S136–S144, 2002.

[24] Y. Cheng and G. M. Church, "Biclustering of expression data." *Proc. of the 8th International Conference on Intelligent Systems for Molecular Biology*, vol. 8, pp. 93–103, 2000.

[25] Y. Kluger, R. Basri, J. T. Chang, and M. Gerstein, "Spectral Biclustering of Microarray Data : Coclustering Genes and Conditions," *Genome Research*, vol. 13, pp. 703–716, 2003.

[26] H. Turner, T. Bailey, and W. Krzanowski, "Improved biclustering of microarray data demonstrated through systematic performance tests," *Computational Statistics & Data Analysis*, vol. 48, no. 2, pp. 235–254, 2005.

[27] S. Hochreiter, U. Bodenhofer, M. Heusel, A. Mayr *et al.*, "Fabia: factor analysis for bicluster acquisition." *Bioinformatics*, vol. 26, no. 12, pp. 1520–7, 2010.

[28] J. Ihmels, G. Friedlander, S. Bergmann, O. Sarig, Y. Ziv, and N. Barkai, "Revealing modular organization in the yeast transcriptional network." *Nature genetics*, vol. 31, no. 4, pp. 370–7, 2002.

[29] D. T. Truong, R. Battiti, and M. Brunato, "Discovering non-redundant overlapping biclusters on gene expression data," in *ICDM 2013*. IEEE, 2013, pp. 747–756.

[30] The Cancer Genome Atlas Network, "Comprehensive molecular portraits of human breast tumours." *Nature*, vol. 490, no. 7418, pp. 61–70, Oct. 2012.

[31] Q. Mo, S. Wang, V. E. Seshan, Olshen *et al.*, "Pattern discovery and cancer gene identification in integrated cancer genomic data." *PNAS*, vol. 110, no. 11, pp. 4245–50, 2013.

[32] B. Wang, A. M. Mezlini, F. Demir, M. Fiume, Z. Tu, M. Brudno, B. Haibe-Kains, and A. Goldenberg, "Similarity network fusion for aggregating data types on a genomic scale." *Nature methods*, vol. 11, no. 3, pp. 333–7, 2014.

[33] M. Goldman, B. Craft, T. Swatloski, K. Ellrott, M. Cline, M. Diekhans, S. Ma, C. Wilks, J. Stuart, D. Haussler, and J. Zhu, "The ucsc cancer genomics browser: update 2013," *Nucleic Acids Research*, vol. 41, no. D1, pp. D949–D954, 2013.

[34] C. H. Mermel, S. E. Schumacher, B. Hill, M. L. Meyerson, R. Beroukhim, and G. Getz, "GISTIC2.0 facilitates sensitive and confident localization of the targets of focal somatic copy-number alteration in human cancers." *Genome biology*, vol. 12, no. 4, 2011.

[35] J. S. Parker, M. Mullins, M. C. U. Cheang, S. Leung *et al.*, "Supervised risk predictor of breast cancer based on intrinsic subtypes." *Journal of clinical oncology*, vol. 27, no. 8, pp. 1160–7, Mar. 2009.

[36] S. Lin, "Rank aggregation methods," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 2, no. 5, pp. 555–570, 2010.

[37] F. Rossi, K. B. Venable, and T. Walsh, "A short introduction to preferences: Between artificial intelligence and social choice," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 5, no. 4, pp. 1–102, 2011.

[38] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar, "Rank aggregation methods for the Web," *Proc. of the 10th International Conference on World Wide Web (WWW-01)*, pp. 613–622, 2001.

[39] T. B. Murphy and D. Martin, "Mixtures of distance-based models for ranking data," *Computational Statistics & Data Analysis*, vol. 41, no. 3–4, pp. 645 – 655, 2003.

[40] P. Awasthi, A. Blum, O. Sheffet, and A. Vijayaraghavan, "Learning mixtures of ranking models," in *Advances in Neural Information Processing Systems (NIPS-14)*, vol. 3, 2014, pp. 2609–2617.

[41] W. Ding, P. Ishwar, and V. Saligrama, "A topic modeling approach to ranking," in *Proc. of the 18th International Conference on Artificial Intelligence and Statistics (AISTATS-15)*, vol. 38, 2015.

[42] F. Chierichetti, A. Dasgupta, R. Kumar, and S. Lattanzi, "On learning mixture models for permutations," in *Proc. of the 2015 Conference on Innovations in Theoretical Computer Science (ITCS-15)*, 2015, pp. 85–92.

[43] S. Agarwal, "On ranking and choice models," in *Proc. of the 25th International Joint Conference on Artificial Intelligence (IJCAI-16)*, 2016, pp. 4050–4053.

[44] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society*, vol. 58, pp. 267–288, 1994.

[45] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM Review*, vol. 43, no. 1, pp. 129–159, 2001.

[46] S. Karaev and P. Miettinen, "Capricorn : An Algorithm for Subtropical Matrix Factorization," in *Proc. of the 2016 SIAM International Conference on Data Mining (SDM-16)*, 2016, pp. 702–710.

Thanh Le Van received his Bachelor degree in Computer Science from Ho Chi Minh City University of Technology, his master degree in Information and Communication Technology from Asian Institute of Technology (AIT), Thailand, and his PhD in Computer Science, KU Leuven, Belgium. He is currently a postdoctoral researcher in the MAGNET group, INRIA in Lille, France. His main research interest is declarative approaches to data mining using principles of optimisation and their applications.

Siegfried Nijssen received his master degree and his Ph.D. in Computer Science from the Universiteit Leiden (The Netherlands). He was a postdoc at the Albert-Ludwigs-University Freiburg and the KU Leuven. He was assistant professor at Leiden University in the period 2012-2016. Since 2016 he is assistant professor at the Institute of Information and Communication Technologies, Electronics and Applied Mathematics of the Université catholique de Louvain, Belgium. His research interests include constraint-based data mining, mining patterns in structured data, declarative approaches to data mining and its applications.

Matthijs van Leeuwen is assistant professor in data mining at Leiden University, the Netherlands. He received his Ph.D. in Computer Science in 2010 from Utrecht University, the Netherlands. After that he has been a postdoctoral researcher at Utrecht University, KU Leuven (Belgium) and Leiden University. His main research interest is exploratory data mining, often based on pattern mining in combination with information theory and/or interactive data mining.

Luc De Raedt received his degree in Computer Science from the Katholieke Universiteit Leuven (Belgium) in 1986 and his PhD. in Computer Science from the same university in 1991. From 1999 till 2006 he was a full Professor at the Albert-Ludwigs-University Freiburg. Since then, he has taken up a (full) research professorship (BOF) at the Department of Computer Science of the Katholieke Universiteit Leuven (Belgium), where he joined the lab for Declarative Languages and Artificial Intelligence (DTAI). Luc De Raedts research interests are in Artificial Intelligence, Machine Learning and Data Mining, as well as their applications. His interests are in probabilistic logic learning, the integration of constraint programming with data mining and machine learning principles, the development of programming languages for machine learning, analyzing graph and network data, and their applications in real-life applications.